
OCR
Oxford Cambridge and RSA
An OCR endorsed
teaching and learning tool
OCR
A Level
Computer
Science
H446 – Paper 1
4
Floating point
arithmetic
Unit 6
Data types
PG ONLINE

Objectives

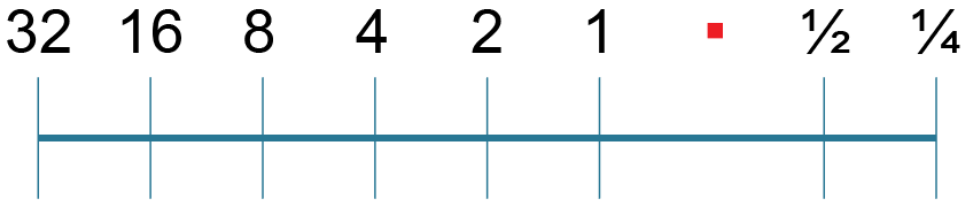
- Represent positive and negative numbers with a fractional part in floating point form
- Normalise un-normalised floating point numbers with positive or negative mantissas
- Add and subtract floating point numbers

Fixed point

- Fixed point binary numbers have a pre-determined number of bits before and after the point
- This makes fixed point numbers easier to process but they cannot represent the **range** or **accuracy** of numbers that may be required

Fixed point binary numbers

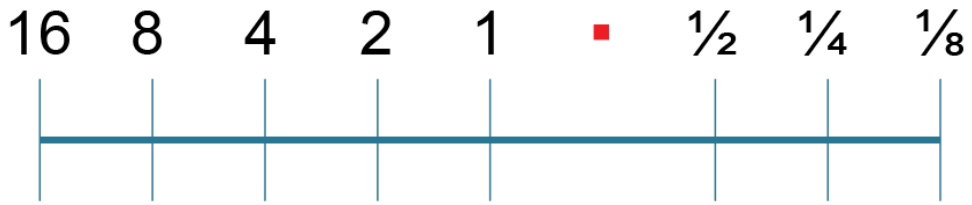
- We could hold an 8-bit number in fixed point format like this:



- If the first bit is a sign bit representing -32, what is the largest positive number that can be held?
- If we want to hold larger numbers, the point must be set further to the right
  - What effect does this have on accuracy?

Fixed point binary numbers

- Suppose you want to convert the decimal number 4.6 to binary using this representation



- The closest you can get is 00100.100, i.e. 4.5
- There is a rounding error of 0.1.
- If you need more accuracy, the binary point would need to be moved left

Floating point numbers

- When decimal numbers become very large, they are held in the format m x 10^n where m is known as the mantissa, and n is the exponent
- The number 75,000 can be represented as 0.75 x 10^5
- How would you represent 458.675?
- How could you represent 0.005 in this format?

Floating point numbers

- When decimal numbers become very large, they are held in the format m x 10^n where m is known as the mantissa, and n is the exponent
- The number 75,000 can be represented as 0.75 x 10^5
- How would you represent 458.675? This could be represented as 0.458675 x 10^3 or 4.58675 x 10^2, for example
- How could you represent 0.005 in this format? This could be represented as 0.5 x 10^{-2} or 5 x 10^{-3}

Floating point numbers

- Binary numbers are typically held in 32 or 64 bits
- In our examples we will use just 12 bits, with the first bit being a sign bit

Sign bit | Mantissa | Exponent

0 • 1 0 1 1 0 1 0 0 0 1 1

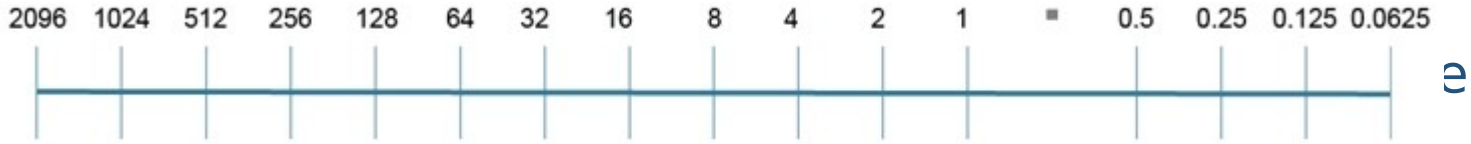
- The number above represents the number

  0.1011010 x 2^3 since the exponent is 3 in decimal

- This is 101.1010 or 4 + 1 +0.5 + 0.125 = 5.625

Floating point binary to denary

e.g. convert 0.1101100 0100 to denary, assuming an 8-bit mantissa and 4 bit exponent

- Write down the mantissa, 0.1101100
- Translate the exponent from binary to decimal; 0100 = 4
- The binary point of the mantissa has to be moved 4 places right to multiply it by 2^4
- 

e below.

- Answer 13.5

Negative binary to denary

- A negative floating point number will start with a 1 (the sign bit)
- The number 1.0010100 0011 has a negative mantissa and a positive exponent
  - Find the two's complement of the mantissa
  - Translate the exponent to denary, giving 3
  - The binary point has to be moved 3 places right to multiply it by 2^3
  - What is the binary number and its decimal equivalent?

Negative binary to denary

- Here's the answer:
- The number 1.0010100 0011 has a negative mantissa and a positive exponent
  - The mantissa is 1.0010100
  - The two's complement of 1.0010100 is 0.1101100
  - The exponent is 3
  - The mantissa, with the binary point moved 3 places right, is 0110.1100
  - The number is -6.75

Negative mantissa

- Try this one:

1 . 0101011 0101

  - Find the two's complement of the mantissa (0.1010101)
  - Translate the exponent to decimal giving 5
  - Move the binary point 5 places right
  - Do you get an answer of -21.25?

Negative exponent

- The leftmost bit of the exponent is a sign bit, just as it is in the mantissa
- An exponent of 1000 in a 4-bit exponent is -8
- You can calculate the denary value of a negative exponent, for example 1011, as -8 + 3 = -5
- Or, you can use the twos complement method which will give you the same result
- What is the denary value of negative exponent 1111?

Negative exponent

- If the exponent is negative, the binary point must be moved left instead of right

  e.g. convert 0.1100000 1110 to denary, assuming an 8-bit mantissa and 4 bit exponent

  - Write down the mantissa, 0.1100000
  - Translate the exponent from binary to decimal; 1110 = -2
  - The mantissa has to be moved 2 places left to divide it by 2^2
  - Binary number is 0.0011000 = 0.125 + 0.0625 = 0.1875

Worksheet 4

- Now try the questions in **Task 1** of the worksheet

Normalisation

- Normalisation is the process of moving the binary point of a floating point number to provide the maximum level of precision for a given number of bits
- This is achieved by ensuring that the first digit after the binary point is a significant digit

Normalisation

- Here are some examples in denary:

$$0.1234567 \times 10^{7} = 1,234,567$$

$$0.0123456 \times 10^{8} = 1,234,560$$

$$0.0012345 \times 10^{9} = 1,234,500$$

- The first representation with a significant (non-zero) digit after the decimal point has the maximum precision using the same number of digits

Normalisation

- In normalised floating point form:

  A positive number has a sign bit of 0 and the next digit is always 1

- This means that the mantissa of a positive number in normalised form always lies between ½ and 1

Example

- Normalise the binary number 0.0011010 0100
  - The binary point needs to move 2 places to the right so that there is a 1 following the binary point
  - Pad with 0s at the righthand end, giving 0.1101000
  - To compensate for making the mantissa larger, we must make the exponent smaller
  - Subtract 2 from the exponent making it 0010
  - The normalised number is 0.1101000 0010

Try this one:

- Normalise the binary number 0.0000111 0111

Answer to example

- Normalising the number 0.0000111 0111 means moving the binary point 4 places right, and subtracting 4 from the exponent
- This gives 0.1110000 0011

Normalised negative numbers

- An unnormalised negative number will have a sign bit of 1 and one or more 1s after the binary point
- e.g. 1.1100011 0011
- A normalised negative number has a sign bit of 1 followed by a zero after the binary point

Normalising a negative number

Example: normalise the number 1.1100011 0011

- Move the binary point 2 places right and subtract 2 from the exponent

  111.00011 0001

- This is the same as 1.0001100 0001
- *Note that leading 1s in a negative number do not change the value of the number (like leading 0s in a positive number)*

From denary to normalised binary floating point

Eg: Convert 88 to normalised floating point binary

- First convert the number to fixed point binary
  - In fixed point binary, the number is 01011000
  - Move the binary point 7 places left
  - Set the exponent equal to 7
- The number is 0.1011000 0111

Exercise: Now try converting 17.25 to normalised floating point binary

Answer to exercise

- Convert 17.25 to normalised floating point binary
  - Convert to fixed point binary 10001.010
  - Move binary point 5 places left
  - Set exponent to 5
- Binary number is 0.1000101 0101
- Carry out the opposite conversion to check the result

Converting a negative denary number

- If the number is negative, calculate the two's complement before normalising
- e.g. calculate the binary equivalent of -17.75
  - The number is `(-) 010001.11`
  - One's complement `101110.00`
  - Two's complement `101110.01`
  - Move the point 5 places left
  - Set the exponent equal to 5
- The number is 1.0111001 0101

Worksheet 4

- Now try **Task 2** on the worksheet

Adding floating point numbers

- First, consider two denary numbers with different mantissas, 1534×10^3 and 1025×10^2
- Clearly we cannot simply add 1534 and 1025, since these numbers represent 1,534,000 and 102,500
- Similarly, we cannot add the mantissas of two floating point binary numbers until we have equalised the mantissas
- To equalise the mantissas, convert the numbers to fixed point binary

Example

- Add the two floating point numbers

```
A = 0.1100000  0001    B = 0.1111100   0011
```

- Convert to fixed point and add:

```
      A =       1.1000
      B =     111.1100
    Sum =    1001.0100
```

- The number is positive, so sign bit will be 0
- Convert to normalised floating point:

```
Result = 0.1001010   0100
```

Subtracting floating point numbers

- To subtract a floating point number from another, first convert them both to fixed point
- Find the two's complement of the number to be subtracted
- Add the two numbers
- Convert result to normalised floating point

Example

- Subtract the two floating point numbers, A - B

```
A = 0.1101000  0100    B = 0.1110000   0011
```

- Convert to fixed point:

```
A =   1101.0000

B = - 0111.0000 (one's complement =
1000.1111)
```

- Find two's complement of B: `1001.0000`
- Add A + (-B): `0110.0000` (ignore overflow bit)
- Convert back to normalised floating point:

```
Result = 0.1100000   0011
```

Worksheet 4

- Now try **Task 3** on the worksheet

Plenary

- Be sure you can:
  - Convert floating binary numbers to decimal and vice versa
  - Normalise floating-point numbers with positive and negative mantissas
  - Explain the effect of increasing the size of the mantissa and exponent
  - Add and subtract floating point numbers

Copyright

Licence agreement